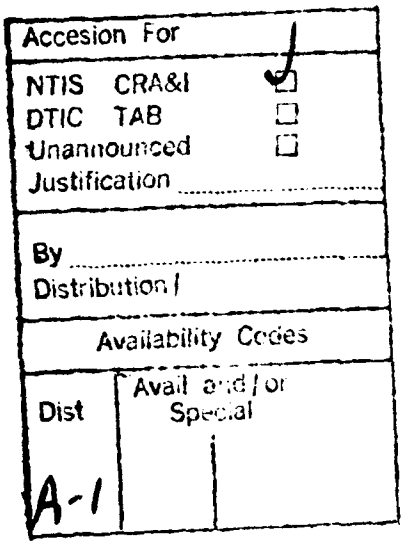
Principal Investigator Name: Michael W. Mislove
PI Institution: Tulane University
PI Phone Number: (504) 865 - 5727
PI E-mail Address: mwm@tulmath.math.tulane.edu
Grant of Contract Title: A Uniform Approach to the Semantics of Concurrency
Grant or Contract Number: N00014-88-K-0499
Reporting Period: 1 July 88 - 31 Dec 91

This constitutes the Final technical Report for ONR Research Contract No. N00014-88-K-0499, entitled, *A Uniform Approach to the Semantics of Concurrency.* The format of the report is the same as is used for annual reports to ONR to report progress on continuing contracts and grants.

Principal Investigator Name: Michael W. Mislove
PI Institution: Tulane University
PI Phone Number: (504) 865 - 5727
PI E-mail Address: mwm@tulmath.math.tulane.edu
Grant of Contract Title: A Uniform Approach to the Semantics of Concurrency
Grant or Contract Number: N00014-88-K-0499
Reporting Period: 1 July 88 - 31 Dec 91

1) Productivity Measures:
Refereed papers submitted but not yet published: 2
Refereed papers published: 7
Unrefereed reports and articles: 7
Books or parts thereof submitted but not yet published: 0
Books or parts thereof published: 5
Patents filed but not yet granted: 0
Patents granted: 0
Invited presentations: 31
Contributed presentations: 9
Honors received: 11
Prizes or awards received: 3
Promotions obtained: one
Graduate students supported $\geq 25\%$: 0
Post-docs supported $\geq 25\%$: 0
Minorities supported: 0



Statement A per telecon James Smith
ONR/Code 1267
Arlington, VA 22217-5000

NWW 2/25/92

Principal Investigator Name: Michael W. Mislove
PI Institution: Tulane University
PI Phone Number: (504) 865 - 5727
PI E-mail Address: mwm@tulmath.math.tulane.edu
Grant of Contract Title: A Uniform Approach to the Semantics of Concurrency
Grant or Contract Number: N00014-88-K-0499
Reporting Period: 1 July 88 - 31 Dec 91

2) Detailed Summary of Technical Progress:

Description of the Scientific Goals The role of semantics is to allow reasoning about high-level programming languages through the use of models. Our research focuses on two approaches to semantics, Denotational Semantics and Operational Semantics, and their applications to languages supporting concurrent computation. In Denotational Semantics, mathematical models are studied which have the same operations on them as those that are available in the language (such as sequential composition and parallel composition, etc.) and a function is defined which assigns meanings in the model to programs in the language. This function should be *compositional*, i.e., a homomorphism with respect to the operations of the language. In this way, the meaning of a complicated program is made up as the composite of the meanings of its constituent parts, so the meanings of complicated programs can be understood in terms of the meanings of their simpler constituents.

The other approach we study is Operational Semantics, where the meaning of a program is given by means of a Labelled Transition System which reflects the transitions of the program on an "abstract" machine. In this approach, an idealized implementation of the program is developed, and problems associated with implementation can be addressed in an abstract setting. This approach provides a "real-world" (albeit idealized) counterpart to the more abstract denotational model. There are a variety of ways in which the transition system is utilized to study the language. In one, a *behavior* is assigned to each program. This can be the set of all sequences of transitions the program undergoes in the system, or only those which represent the completed sequences of transitions. Or, in studying languages based on state transition functions, one can assign the set of states which the program passes through during its "execution," or the final state, etc. as the behavior.

In studying a given language, it is important to utilize both of these approaches, and to relate the semantic models. The standard terminology in this regard is as follows. A denotational model is said to be *adequate* with respect to an operational model if whenver two programs have the same meaning in the denotational model, then they have the same behavior in the operational model. Dually, the denotational model is said to be *fully abstract* with respect to the operational model if, whenever two programs have different meanings in the denotational model, then one can find an "environment" in which, once placed, the programs exhibit different behaviors in the operational model. The goal, then, is to discover operational and denotational models for languages for which the denotational model is adequate and fully abstract with respect to the operational model. Such a pair of models allows both an abstract understanding of the language in the denotational model, and a concrete understanding in the operational model. One can then analyze programs in the more abstract model with the confidence that the results obtained have implications for

the real-world behavior of the program. In particular, denotational models are particularly amenable to the study of such questions as safety and liveness, and one can craft logics for the denotational model which address these concerns.

Our work in this contract addresses a particularly important area of denotational semantics, the semantics of concurrency. Languages which support concurrent computation are becoming more and more prevalent as concurrent machines become available, and there is a fairly large body of research already in place on the semantics of concurrency. However, different authors have studied different languages and varying semantic models for those languages and their constructs, and so it is difficult even to begin to compare the various features these languages support. A first step in carrying out such a comparison is to find a uniform framework for the semantic models for these languages, and to utilize this framework as a point of reference for analyzing other approaches. Among models presented for concurrency, there are domain-theoretic models which were built up from the very successful domain-theoretic models for sequential deterministic languages. There are also metric space models, based on the more widely understood concepts of metric spaces and continuous functions. And there are ordered structures which have been used as semantic models, most notably in the work by Hoare and his school on *CSP* (*cf.* [1, 2]). There are other models as well.

In our approach to studying concurrent languages, we start with the usual domain-theoretic constructs and develop new constructs and tools which simplify the modeling process, and which make the various features of the many languages easy to analyze and compare. We have chosen a domain-theoretic approach because it supports a particularly wide range of features in a uniform way. There is also an extensive literature about domains and their use in modeling languages to draw on. Of particular interest are the three basic power domain constructs which have been utilized for building models for concurrent languages from models for sequential, deterministic languages. In fact, our work has come to focus on this aspect, and our principal results address improvements in this modeling technique. The results of our work can be summarized as

i) providing a clearer understanding of the relationship between determinism and nondeterminism, at least in the case of *angelic nondeterminism*,

ii) extending results about the Hoare power domain to the class of algebraic posets, which in turn allows a simpler explanation of how particular models can be built, and

iii) extending the modeling technique to include the use a finer topology on the semantic model than has historically been utilized. In analogy to the case of complete metric space models, this topology is compact and Hausdorff, so that limit points are unique when they exist. This last aspect offers the promise of more powerful proof techniques for reasoning about programs in terms of the model.

In each case, we demonstrate the results we cite using a simple language by generating a denotational model which is adequate and fully abstract with respect to an associated operational model.

The basis of our work has been to refine the use of the Scott topology (or, more precisely, the family of Scott closed subsets of the domain) as a power domain construct. The family of Scott closed sets in the usual containment order is well known as the Hoare power domain, and it was shown early on that this construct leads to a left adjoint to

the forgetful functor from a category of domains supporting a continuous nondeterministic choice operator to the category of domains and Scott continuous maps. We have developed a duality theory for domains using the spectral theory of completely distributive algebraic lattices which amplifies this adjunction for the Hoare power domain.

First, the usual Hoare power domain is based on building a model for nondeterminism from a model for determinism. Our theory, which associates to a completely distributive algebraic lattice its family of sup-primes, provides a method to go back. The family of sup-primes of a completely distributive algebraic lattice is a domain, and this association gives rise to a right adjoint to the functor which associates to a domain its family of non-empty Scott closed subsets. In fact, we show that these functors form an equivalence of categories. The utility of this work in semantics is based on the idea that a language supporting nondeterminism and concurrent computation has an underlying sequential, deterministic language which generates the full language using the operators of the language. So, while the usual Hoare power domain is an example of how one can build a model for a concurrent language from one for the underlying sequential, deterministic sublanguage, our theory allows one to "go back," and to recognize the model of the sequential, deterministic sublanguage from within the larger model. Moreover, since the functors we are using form an equivalence, the model for the sequential, deterministic sublanguage completely determines that for the larger language, just as the sequential, deterministic sublangauge generates the full language.

Our theory also shows how to construct a natural semantic model for a language from the syntax of the language, and it shows what assumptions on the syntactic operators of the language will lead to well-behaved (i.e., continuous) operators in the semantic model; it provides a mechanism for defining each semantic operator from the corresponding syntactic one. We have applied this theory to the develop a new abstract language which utilizes the rendezvous mechanism for synchronization among several processors; this mechanism is used, e.g., in the Ada programming language. We have given an operational semantics for our language based on a simple transition system utilizing rewrite rules and algebraic equations. We have also developed a denotational model for the language, and shown that this model captures just the right level of abstraction for the operational model of the language.

Our work, which is being done in collaboration with Dr. Frank J. Oles (IBM Thomas J. Watson Research Center), has focused on the problem of giving a natural operational and denotational semantics for the abstract language whose syntax is given as follows:

$$p ::= a \mid \texttt{triv} \mid \texttt{undef} \mid p;p \mid p+p \mid p\|p \mid \texttt{trec}[p,q],$$

where `triv` is the process whose sole action is immediate normal termination, `undef` denotes immediate abnormal termination, ; denotes sequential composition, + denotes nondeterministic choice, $\|$ denotes parallel composition (via interleaving), and `trec` represents a simple tail-recursion construct. This language is inspired by one considered in the seminal paper of Hennessy and Plotkin, *Full abstraction for a simple parallel programming language* (**LNCS 74** (1979)). It is important to understand that the elements $a \in A$ represent *autonomous atomic actions*, in that these actions occur without the participation of the environment; this is in contrast to the atomic actions considered in the language CSP, for example.

The operational semantics for our language is given via a transition system which shows how an abstract machine would step through a given program in our language. These systems are usually presented as a long, exhaustive list of transition rules which apply to any program in the language. With such a system, it becomes very difficult to understand the result of applying the transition rules to a given program. Our approach has been to break up the transition system into two sets of rules: *operational rewrite rules* and *transition rules*. The first set of rules, the operational rewrite rules, is designed to "percolate to the top" the next atomic action to be performed in a program. Most of the complication of the transition system is contained within this set of rules, and the principal result about them is that they are strongly normalizing and Church-Rosser. Thus, each program has a unique normal form with respect to the rewrite rules, and this form is reachable in finitely many rewrite steps. That is, the rewrite rules are computationally feasible.

With most of the complication of the transition system confined to the rewrite rules, the transition rules for our transition system are remarkably simple; they encompass less than a dozen rules. This makes analysis of their affect on a program much simpler than with the transition systems one usually encounters. We have chosen to implement *angelic nondeterminism* in our semantic model. This is reflected in the fact that the nondeterministic sum of two programs p and q is their least upper bound in the denotational model. It is realized in the transition system by the following rules, which govern the transitions that can be made from a term of the form $p + q$:

$$\frac{p \xrightarrow{a} q,\ r \xrightarrow{a} s}{p + r \xrightarrow{a} q + s} \qquad \frac{p \xrightarrow{a} q,\ r \not\xrightarrow{a}}{p + r \xrightarrow{a} q} \qquad \frac{p \not\xrightarrow{a},\ r \xrightarrow{a} s}{p + r \xrightarrow{a} s}.$$

There are two advantages to using this form of nondeterminism. First, processes become *reliable*, in that a process which executes a given string of actions and then deadlocks on one run, cannot execute the same string of actions on another run and then continue on to do other actions. The second advantage is related to the first: it is the fact that the set of maximal strings of actions which a process can perform - what we call the *behavior* of the process - is an antichain in the set $A^* \cup A^*\surd \cup A^\infty$ of all finite or infinite strings over the alphabet A of atomic actions, where $\surd$ is a special symbol which denotes normal termination. Moreover, the lower set of the behavior of a program is Scott closed, a result which allows us to use the non-empty Scott closed subsets of the domain $A^* \cup A^*\surd \cup A^\infty$ as a denotational model for our language. This family is the Hoare power domain over $A^* \cup A^*\surd \cup A^\infty$, but there are two essential differences in our use of it as compared to the usual use that is made of this power domain.

The first difference in our approach is that we equip the Hoare powerdomain with a topology which is finer than the Scott topology. This finer topology, the *Lawson topology*, is compact and Hausdorff (whereas the Scott topology is only T_0), so limits in this topology are unique. To use this stronger topology, one must verify that the functions which are needed in the denotational model are Lawson continuous. Not only have we been able to do this, we have developed a theory of continuous posets which delineates exactly how the Lawson continuity of the functions needed for the denotational model can be guaranteed by conditions on the operations of the language. That is, our theory characterizes in simple

terms what conditions the operations which are given in the syntax of the language must satisfy in order that there be Lawson continuous extensions of these operations on the denotational model.

The second difference in our use of the Hoare power domain stems from our use of the spectral theory of locally compact sober spaces. We use spectral theory both as a basis for relating our theory of algebraic posets to the more established theory of domains and to derive a natural definition of the denotational model for our language. We first use spectral theory to prove that any algebraic poset has a completion to a domain in which the crucial way-below relation is determined by the way-below relation on the underlying algebraic poset. The completion to which we refer is simply the sobrification of the underlying poset in the Scott topology. This result gives rise to a left adjoint to the embedding functor from the category of domains and Scott continuous maps into the corresponding category of algebraic posets. Actually, there is a second adjunction here: the same functor has a restriction which is the left adjoint to the embedding functor from the category of algebraic cpo's and monotone Lawson continuous maps into a corresponding category of algebraic posets. Viewed in this way, these results provide a topological alternative to the well-known adjunction which says that each domain is the ideal completion of its set of compact elements.

Our use of spectral theory has a second application to semantics. We associate to an algebraic poset its family of nonempty Scott closed subsets endowed with the Lawson topology. According to our theory, given a denotational model of a deterministic language where all the functions involved are Lawson continuous, the natural denotational model for the language endowed with angelic nondeterminism is the family of non-empty Scott closed subsets of the determninistic model. Moreover, spectral theory applies to show that the deterministic processes can be singled out in this model as the the set of $\cup$-primes in the Scott closed sets. This allows us to go back and forth between deterministic models and nondeterministic models.

Finally, the denotational model we present for angelic nondeterminism is adequate and fully abstract with respect to the operational model for the language dscribed above. Indeed, while the proof that the operational rewrite rules are strongly normalizing and Church-Rosser is rather detailed and involved, the proof that the denotational model is adequate and fully asbtract is very simple and elegant. We believe this is a major contribution of this line of research. Most proofs of full abstraction, even for simple languages, are quite arcane and difficult to follow. On the other hand, our proof is very staightforward and easy to follow. This convinces us that we have found the correct way to give operational and denotational models for languages such as the one we are considering. These results have been presented both in numerous colloquia, and as part of last spring's MFPS meeting (cf. [6]). A major paper on these results ([7]) has just been completed and submitted to *Theoretical Computer Science*. In future work, we intend to examine the remaining forms of nondeterminism, *demonic nondeterminism* and *conventional nondeterminism*, with the hope of obtaining results analogous to those in [6,7].

Finally along this line, we have also devised an enhancement to our language which uses the rendezvous protocol for synchronization between processes. This protocol is the basis for communication and synchronization in Ada©; it is like a remote procedure call.

in that pairs of processes synchronize when one of them has a task which it wishes the other to perform for it. We have embellished the protocol by allowing both the calling process and the process being called to contribute a portion of the critical region. A BNF-like set of production rules for RSP is the following:

$$p ::= a \mid \texttt{triv} \mid \texttt{undef} \mid \sigma[p,p] \mid p : p \mid p + p \mid p \| p \mid \texttt{trec}[p,p],$$

where $a \in A$ varies over the set of atomic actions which processes p can perform. These atomic actions are intended to represent actions which a process can execute on its own, without the cooperation of the environment. Thus, as opposed to the synchronization events which make up CSP, for example, the actions we use are autonomously executed by any process in the language.

Synchronization between processes in RSP is accomplished using a generalization of the rendezvous mechanism. The process $\sigma[p,q]$ denotes a program which wishes to rendezvous with the complementary process $\bar{\sigma}[r,s]$. So, the set S of *synchronization operators* has an involution $\sigma \mapsto \bar{\sigma}$. Once the rendezvous takes place, for example when we have a process $\sigma[p,q] \| \bar{\sigma}[r,s]$, the critical region is composed of the process $(p \| r) \| (q \| s)$. The reason that there are two components, p and q, in the body of each synchronization event σ is a consequence of our intuition of how distinct rendezvous should relate to one another. Having only one component does not appear to be strong enough to allow distinct rendezvous to overlap one another in terms of their beginning and end.

The remaining operators are the same as for the simpler language without synchronization described earlier: ; denotes sequential composition, + denotes nondeterministic choice, $\|$ denotes parallel composition, and $\texttt{trec}[p,q]$ denotes a form of linear tail recursion.

Work on RSP on the implementation side also progressed during the period of this contract. The formality and high-level of abstraction of RSP can be used effectively to specify and reason about concurrent systems. Properties such as safety and liveness can be proved in a rigorous way. If the concurrency constructs provided by the implementation language are based on the same abstractions as the design language, the correctness of the implementation will follow directly from the formal design. We have chosen RSP as the formalism for specifying, designing and reasoning about concurrency, and Ada as the implementation language. A systematic translation method (to be formalized later) from RSP to Ada will ensure that properties proved for RSP will hold in the Ada context. We have developed algorithms to support this translation process. A front-end processor (lexical analysis, syntactic analysis, and intermediate code generation) has been implemented.

Because an RSP process is defined algebraically the execution state of an RSP process can be represented as a tree of processes. At the leaves of the tree are the primitive RSP operators such as prefix, input and output; interior nodes are the process combinators such as parallelism and choice. A simple execution model based on the tree representation consists of a repetition of a bottom-up computation of the initials sets, followed by a top-down engagement in one of the events in the initials sets. For each leaf node the initials set contains the process' event argument. Each interior node has specific rules for computing initials sets based on the initials sets of its operands. For the top-down execution of an event there is a simple rule: a process engages in the event only if that event is in its initials set. This rule does not ensure that every process prepared to engage in the event

will actually do so; several RSP operators, such as general-choice have more selective rules for propagating events downward.

In the execution model described above the complete behavior of leaf node processes was determined at compile-time. The question we need to address is whether or not it is possible and efficient to determine the complete behavior of the RSP combinator operations at compile time. In the general case it is not possible to reduce an arbitrary RSP process into a simple canonical form. A proof that CSP is not reducible can be found in Hoare and Olderog's paper [3]; because of this, we do no expect RSP to be reducible either. For some processes a reduction to a simple canonical form is possible. However, these reductions result in an execution model that is equivalent to the model presented above with optimizing tree transformations. Using rewriting rules it is possible to expand an RSP combinator operation, but the code size and complexity increases dramatically. Code expansion via rewriting is not appropriate in our overall approach because we intend that the programmer will read and modify the generated Ada code.

Two different approaches for translating RSP were investigated. The first approach allows the computation of the initials of a process to be performed dynamically and on demand only. The second approach synthesizes statically the initials of all the processes. In either case, the inputs to the translation algorithms are an abstract syntax tree and a symbol table. Leaf nodes in the tree are viewed as primitive or independent processes, and internal nodes represent compound processes.

The work described so far represents the first part of our proposed three-part program. We are also working on devising appropriate operational and denotational models for demonic and conventional nondeterminism. We have made progress on both the operational and denotational models for demonic nondeterminism. The operational model is very similar to the one for angelic nondeterminism, and we expect the analysis of this model to be analogous to the one for the angelic case. In work with a Ph.D. student of mine, a generalization of the Smyth power domain has been established which works for algebraic posets. We expect that the appropriate category to use for the denotational model for demonic nondeterminism will soon emerge from this work. Here again, our goal is to establish a structure for demonic nondeterminism similar to the one we found for angelic nondeterminism. In particular, we expect to find a denotational model which is generated by elements which are analogous to the sup-primes in the angelic case, and these should be the meanings of deterministic processes. This would allow passing back and forth between models for the deterministic sublanguage and the model of the full language including nondeterminism.

The principal investigator spent January - August, 1991 on sabbatical leave at Oxford University, collaborating with Drs. G. M. Reed and A. W. Roscoe and their students. A full description about those activities is contained in the report on the Foreign Travel Support which was provided for this sabbatical trip, and we will confine our comments here to a brief outline of the collaboration which has resulted from the visit.

An important result of the visit is the paper [6], which gives a general mathematical theory for deriving denotational models supporting unbounded nondeterminism in Timed CSP and untimed CSP. The model for untimed CSP was originally derived by Roscoe, and then a corresponding model for Timed CSP was derived by Schneider. In both timed

and untimed CSP modeling unbounded nondeterminism necessarily leads one to consider spaces which are neither complete partial orders (which were used to model untimed CSP), nor complete metric spaces (which were used to model timed CSP). What was first discovered in our collaboration was that the theory which Roscoe used to derive his model and that which Schneider was using to derive his were instances of a more general mathematical theory which involves *local cpo semilattices*. These are inf-semilattices in which only those directed sets having some upper bound are guaranteed to have a least upper bound. One then uses a theory of dominating functions defined on smaller spaces where they are guaranteed to have fixed points to guarantee that all the operators from the language have meanings in the model defined via least fixed points. This approach provides a completely abstract validation that recursive operators from the language have meanings in the denotational model which are given by least fixed points, but these fixed points can take more than ω iterations to be reached. In an interesting turn of events, the related operational model then provides the justification that this least fixed is the computationally correct one, since it agrees under the congruence theorem with the one the operational model defines.

Another notable result of our research under this contract was joint work with Larry Moss and Frank Oles (IBM). This research focused on the relation between Peter Aczel's universe of non-well-founded sets and domain theory. As Aczel himself pointed out, it seems initially very appealing that there should be a close connection, but it proved fairly elusive to pin down. Non-well-founded set theory extends the the usual ZF set theory by replacing the Axiom of Regularity (which disallows sets with infinite descending membership chains) with an alternative. Aczel's theory allow sets which can have infinite descending membership chains, but his theory is a conservative extension of ZF, and has a class model within ZF. The prototypical non-well-founded set is the set having only one member - itself.

It seems attractive that there should be a relation between non-well-founded sets and well-founded sets which is analogous to the relationship between arbitrary elements of a domain and compact elements. In fact, our research, reported in [4] shows this is the case. We construct a domain $\mathcal{C}$ which is a system (in Aczel's sense), and which contains a copy of the herediatrily finite portion of Aczel's non-well-founded universe among the maximal elements. Since $\mathcal{C}$ is a domain, every element is the supremum of compact elements, and we show that the system map from Aczel's hereditarily finite universe maps the well-founded sets onto the maximal elements which are compact. This means that $\mathcal{C}$ is a model of an extended set theory in which every set is the supremum of well-founded objects - the compact elements. Only (some of) the maximal elements of $\mathcal{C}$ represent actual sets. The rest we call *partial sets*, and we propose a theory for this model in the paper [5].

Our results parallel those of Abramsky (which were never written up for publication). In particular, we show that the domain $\mathcal{C}$ satisfies a domain equation, which, naturally enough, involves the Plotkin power domain. But, while Abramsky begins with a solution to a similar equation and uses it to generate a model of the hereditarily finite portion of Aczel's universe, our approach is to constuct the model directly. An interesting unresolved question is whether the two solutions are the same.

In another area, we explored rather completely the question of when the embedding

functor from a category of Scott domains into the category of $SFP-$objects has a left adjoint. This research began when we used spectral theory to devise a functor from $SFP-$objects into Scott domains. This functor turned out to be a left adjoint to the embedding functor from the category of Scott domains and maps preserving all suprema (not just directed suprema) into the category of $SFP-$objects and Scott continuous maps. We also found other such adjunctions: for example, we realized the Smyth powerdomain as a left adjoint to the embedding functor from the category of Scott domains and Scott continuous maps preserving finite infima into the category of $SFP-$objects. Perhaps the most important result of this work is the proof that there is no left adjoint to the embedding functor from the category of Scott domains and Scott continuous maps and the category of $SFP-$objects and Scott continuous maps.

References

[1] Brookes, S. D., C. A. R. Hoare and A. W. Roscoe, "A theory of communicating sequential processes," JACM **31** (July, 1984), 560-599.

[2] Brookes, S. D. and A. W. Roscoe, "An Improved failures model for communicating processes," Lecture Notes in Computer Science **197** (1985), 281-305.

[3] Hoare, C. A. R. and E.-R. Olderog. Specification-oriented semantics, *Acta Informatica*, 1976.

[4] Mislove, M. W., L. S. Moss and F. J. Oles, Non-well-founded sets modeled as ideal fixed points, *Information and Computation* **93**, pp.16 - 54.

[5] Mislove, M. W., L. S. Moss and F. J. Oles, Partial sets, Proceedings of the Workshop on Situation Theory and Situation Semantics, CSLI Lecture Notes **22** (1991), pp. 117 - 132.

[6] Mislove, M. W and F. J. Oles, A simple language supporting angelic nondeterminism and parallel composition, *Proceedings*, Seventh MFPS Conference, to appear.

[7] Mislove, M. W. and F. J. Oles, A topological algebra for angelic nondetermninism, *Theoretical Computer Science*, submitted.

[8] Mislove, M. W., A. W. Roscoe and S. A. Schneider, Fixed points without completeness, in preparation.

[7] Roscoe, A. W., Unbounded nondeterminism for CSP, preprint.

Principal Investigator Name: Michael W. Mislove
PI Institution: Tulane University
PI Phone Number: (504) 865 - 5727
PI E-mail Address: mwm@tulmath.math.tulane.edu
Grant of Contract Title: A Uniform Approach to the Semantics of Concurrency
Grant or Contract Number: N00014-88-K-0499
Reporting Period: 1 July 88 - 31 Dec 91

3) Publications, Presentations, Reports and Awards and Honors:

a) Refered papers submitted but not yet published:

1. Mislove, M. W. and F. J. Oles, A simple language supporting angelic nondeterminism and parallel composition, *Proceedings*, Seventh MFPS Conference, to appear.
2. Mislove, M. W. and F. J. Oles, A topological algebra for angelic nondetermninism, *Theoretical Computer Science*, submitted.

b) Refereed papers published:

3. B. Belkhouche, L. Lawrence and M. Thadani, "A language evaluation methodology and its application to Ada* and Modula 2," Journal of Pascal, Ada and Modula-2 (June-July, 1988).
4. B. Belkhouche, "Implementation and evaluation of a methodology for stepwise refinement of data," Proceedings of the 21st Hawaii International Conference on Systems Sciences (January, 1988), pp. 764-772.
5. Belkhouche, B., Executable Specification Languages: Design and Implementation Issues, Proceedings of the 10th French-Tunisian Conference (May 1989). (with Michael Baingmann).
6. Michael Mislove, "On the Smyth Powerdomain," Proceedings of the 3rd Workshop on the Mathematical Foundations of Programming Semantics. LNCS **298** (1988), pp. 161-172.
7. Mislove, M. W., Non-well-founded sets modeled as ideal fixed points, *Information and Computation* **93**, pp.16 - 54. (with L. S. Moss and F. J. Oles).
8. Mislove, M. W., Partial sets, Proceedings of the Workshop on Situation Theory and Situation Semantics, CSLI Lecture Notes **22** (1991), pp. 117 - 132, (with L. S. Moss and F. J. Oles).
9. Mislove, M. W., Local product structures on homogeneous continua, Topology and Its Applications 31 (1989), 259-268. (with J. T. Rogers, Jr.)

c) Unrefereed reports and articles:

10. Boumediene Belkhouche, "Development of Ada and PL/1 protypes from abstract specifications," Tulane University Technical Report 88-103. (October 1988).
11. Boumediene Belkhouche and Linda Lawrence, "A semantic model of Ada tasking," September, 1988.
12. Belkhouche, B., Minimal Height Trees, Tulane University Technical Report 88-105. (with Richard Svatos). (October 1988).

13. Michael Mislove, "The Scott topology as a model for concurrency," July, 1988.
14. Michael Mislove and Michael Main, "Research Trends in Formal Semantics: *Report of a Discussion*", June, 1988.
15. Mislove, M. W. and F. J. Oles, Adjunctions between categories of domains, preprint, September, 1990.
16. Mislove, M. W. and F. J. Oles, A simple language supporting angelic nondeterminism and parallel composition, preprint, September, 1990.

d) Books or parts thereof published:

17. Mislove, M. W., Compact semilattices, partial orders and topology, in: *The Analytical and Topological Theory of Semigroups*, K. H. Hofmann, J. D. Lawson and J. S. Pym, editors, DeGruyter Expositions in Mathematics **1** (1990), DeGruyter Publishers, Berlin, pp. 105 - 131.
18. Mislove, M. W., Problems in domain theory and topology, in: *Open Problems in Topology*, G. M. Reed and J. van Mill, editors, North Holland Press, pp. 349 - 370, (with Jimmie D. Lawson).
19. Michael Mislove, *editor*, "Proceedings of the 3rd Workshop on the Mathematical Foundations of Programming Semantics," Lecture Notes in Computer Science **298** (1988), 637pp. (with Michael Main, Austin Melton and David Schmidt).
20. Mislove, M. W., editor, *Proceedings of the Fourth Workshop on the Mathematical Foundations of Programming Semantics*, Special Issue of *Theoretical Computer Science* **70**, No. 1 (1990). (with Michael Main).
21. Mislove, M. W., editor, *Proceedings of the Fifth Conference on the Mathematical Foundations of Programming Semantics,* Lecture Notes in Computer Science **441**, Springer-Verlag (1990), (with M. Main, A. Melton and D. Schmidt).

e) Invited presentations:

1. Boumediene Belkhouche, "Implementation and Evaluation of a Methodology for the Stepwise Refinement of Data," 21st Hawaii International Conference on Systems Sciences, January, 1988
2. Belkhouche, B., Methodes et Langages de Specification, 10th Tunisian-French Seminar, Tunis (May 1989).
3. Belkhouche, B., Semantique Formelle du Rendezvous, Universite Technique et Scientifique d'Alger (June 1989).
4. Michael Mislove, "The Failures Model as a Powerdomain," 4th Workshop on the Mathematical Foundations of Programming Semantics, Boulder, May, 1988.
5. Michael Mislove, "Lectures on Aczel's Non-Well-Founded Universe," IBM Hawthorne Research Center, July, 1988 (4 Lectures)
6. Michael Mislove, "Using Powerdomains to Model Concurrency," IBM T. J. Watson Research Center, August, 1988.
7. Michael Mislove, "Minimal Models for the Untyped λ-Calculus," Mini-Workshop on Data Flow Analysis, Manhattan, Kansas, June, 1988.
8. Mislove, M. W., History and applications of compact semilattices, Symposium on the Analytical and Topological Theory of Semigroups, Oberwolfach, West Germany, January, 1989.
9. Mislove, M. W., The Scott topology as a model of concurrency, Universität Essen, West Germany, January, 1989.
10. Mislove, M. W., The Scott topology as a model for concurrency, Technische Hochschule Darmstadt, West Germany, January, 1989.
11. Mislove, M. W., Obtaining non-well-founded sets as limit points of well-founded partial sets, University of Pennsylvania, May, 1989.
12. Mislove, M. W., Using Scott closed sets to model concurrency, University of Pennsylvania, May, 1989.
13. Mislove, M. W., Non-well-founded sets as ideal fixed points, Fourth Symposium on Logic in Computer Science, Asilomar, CA, June, 1989 (with Lawrence S. Moss and Frank J. Oles, presentation by F. J. Oles.)
14. Mislove, M. W., Semialgebraic posets and their relation to algebraic posets, Symposium on General Topology and Its Applications, Oxford, England, June, 1989. (presented by Jimmie D. Lawson.)
15. Mislove, M. W., Continuous posets and models of concurrency, Special Session on Applications of Category Theory, *American Mathematical Society* Regional Meeting, Kansas State University, March, 1990.
16. Mislove, M. W., A uniform approach to the semantics of concurrency, ONR Workshop on Languages, Salt Lake City, Utah, July, 1990
17. Mislove, M. W., The category of continuous posets, Sixth Workshop on the Mathematical Foundations of Programming Semantics, Queens University, Kingston, Canada, May, 1990.

18. Mislove, M. W., The category of continuous posets, Summer Topology Conference, Long Island University, C. W. Post Campus, June, 1990.
19. Mislove, M. W., Denotational and Operational Models of Angelic Nondeterminism, Department of Computer Science, Tulane University, September, 1990.
20. Mislove, M. W., The Category of Continuous Posets, Analytic Topology Seminar, Oxford University, January, 1991.
21. Mislove, M. W., Denotational and Operational Models for a Simple Language Supporting Nondeterminism and Parallel Composition, University of Sussex, February, 1991.
22. Mislove, M. W., *RSP*: A Language Supporting Synchronization Using Rendezvous, ESPRIT BRA SPEC Group Meeting, Mook, Holland, March, 1991.
23. Mislove, M. W., Non-well-founded Sets and Domain Theory, Oxford University, April, 1991.
24. Mislove, M. W., Denotational and Operational Models for a Simple Language Supporting Nondeterminism and Parallel Composition, Queen Mary-Westfield College, University of London, April, 1991.
25. Mislove, M. W., *RSP*: A Language Supporting Synchronization Using Rendezvous, University of Cambridge, April, 1991.
26. Mislove, M. W., Using Algebraic Posets to Model Nondeterminism, University of Sheffield, April, 1991.
27. Mislove, M. W., Using Algebraic Posets to Model Nondeterminism, University of Hull, April, 1991.
28. Mislove, M. W., *RSP*: A Language Supporting Synchronization Using Rendezvous, Imperial College of Science and Technology, April, 1991.
29. Mislove, M. W., Domains and the Denotational Semantics of Nondeterminism, Symposium on Semantics and Model Theory, Schloss Dagstuhl, Germany, June, 1991.
30. Mislove, M. W. Non-well-founded Sets and Domain Theory, Technische Hochschule Darmstadt, Germany, June, 1991.
31. Mislove, M. W., Domains and the Denotational Semantics of Nondeterminism, ONR-ESPRIT Meeting, Auch, France, June, 1991.

f) Contributed presentations:

32. Belkhouche, B., Executable Specification Languages: Design and Implementation Issues, 10th Tunisian-French Seminar, Tunis (May 1989).
33. Mislove, M. W., Non-well-founded sets as ideal fixed points, Mini-Workshop on the Mathematical Foundations of Programming Semantics, Tulane University, April, 1989.
34. Mislove, M. W., On using traces to model the semantics of concurrency, Fifth Conference on the Mathematical Foundations of Programming Semantics, Tulane University, April, 1989.
35. Mislove, M. W., Lectures on domain theory, IBM Thomas J. Watson Research Center, July, 1989.

36. Mislove, M. W., The category of continuous posets, Sixth Workshop on the Mathematical Foundations of Programming Semantics, Queens University, Kingston, Canada, May, 1990.
37. Mislove, M. W., The category of continuous posets, Summer Topology Conference, Long Island University, C. W. Post Campus, June, 1990.
38. Mislove, M. W., The Search for Non-cpo Models of the Untyped Lambda Calculus, First Jumelage Meeting, Stanford University, November, 1990.
39. F. J. Oles, A Simple Language Supporting Angelic Nondeterminism and Parallel Composition, Seventh MFPS Conference, Carnegie-Mellon University, March, 1991.
40. Mislove, M. W., Topological Algebras and Nondeterminism, Part I, LMS Symposium on Category Theory and Computer Science, University of Durham, July., 1991.

g) Honors received:

1. Belkhouche, B., Chair IEEE Computer Society Technical Committee on Computer Languages,
2. Belkhouche, B., IEEE Computer Society Conference and Tutorial Board member.
3. Belkhouche, B., Conference Chair of the Third IEEE Computer Society International Conference on Computer Languages.
4. Belkhouche, B., Program Committee Member of the Fifth Mathematical Foundations of Programming Language Semantics.
5. Belkhouche, B., Reviewer for IEEE TSE, Software, and Computer

6 - 8. Michael Mislove, IBM Summer Faculty Appointment, IBM Corporation, Summers, 1989, 1990, 1991

9. Mislove, M. W., Chair, Mini-Workshop on the Mathematical Foundations of Programming Semantics, Tulane University, April, 1989.
10. Mislove, M. W., Co-Chairman, Fifth Conference on the Mathematical Foundations of Programming Semantics, Tulane University, April, 1989.
11. Mislove, M. W., Program Committee member for the Fifth Conference on the Mathematical Foundations of Programming Semantics, Tulane University, April, 1989.
12. Mislove, M. W., Co-chairman, Seventh Workshop on the Mathematical Foundations of Programming Semantics, Carnegie-Mellon University, Pittsburgh, March 25 - 28, 1991.
13. Mislove, M. W., Program Committee, Seventh Workshop on the Mathematical Foundations of Programming Semantics, Carnegie-Mellon University, Pittsburgh, March 25 - 28, 1991.
14. Mislove, M. W., Co-chairman, Eighth Workshop on the Mathematical Foundations of Programming Semantics, Oxford, England, April 6 - 9, 1992

Principal Investigator Name: Michael W. Mislove
PI Institution: Tulane University
PI Phone Number: (504) 865 - 5727
PI E-mail Address: mwm@tulmath.math.tulane.edu
Grant of Contract Title: A Uniform Approach to the Semantics of Concurrency
Grant or Contract Number: N00014-88-K-0499
Reporting Period: 1 July 88 - 31 Dec 91

4) Transitions and DoD Interactions:

As has been indicated at numerous places above, a great deal of the research on which we are reporting is joint work with Dr. Frank J. Oles of the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. Professor Mislove was a Summer Faculty Visitor at Yorktown for three successive summers. In addition, Dr. Oles has been a visitor at Tulane, and at Oxford during Professor Mislove's sabbatical visit there. These visits have made possible the interactions reported here.

We have reported in the research summary on the joint work which has evolved between the Principal Investigator and the group at the PRG, Oxford. As reported there, this work represents applying techniques we have developed to craft new models for the various CSP languages, and to better understand the existing models. Also, as has been indicated at numerous places above, a great deal of the research on which we are reporting is joint work with Dr. Frank J. Oles of the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

5) Software and hardware prototypes: none